View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by East Tennessee State University

Episode 3.11 – Polar and Bipolar Line Coding

Welcome to the Geek Author series on Computer Organization and Design Fundamentals. I'm David Tarnoff, and in this series we are working our way through the topics of Computer Organization, Computer Architecture, Digital Design, and Embedded System Design. If you're interested in the inner workings of a computer, then you're in the right place. The only background you'll need for this series is an understanding of integer math, and if possible, a little experience with a programming language such as Java. And one more thing. Our topics involve a bit of figuring, so it might help to keep a pencil and paper handy.

In Episode 3.10, we looked at the electrical connection between two digital devices and saw two ways we could use a positive voltage and a zero voltage level to code the logic zeros and logic ones of our data. This unipolar scheme had a disadvantage. The capacitance of a conductor could cause the signal voltage levels to drift toward the positive voltage making it difficult to identify when the transmitted signal was supposed to be zero volts.

To solve the drift problem, some physical lines use polar signaling. In this case, the zero volt level is replaced with a voltage that is equal in magnitude and opposite in polarity to the positive voltage. Bipolar signaling is a further extension of polar signaling. Bipolar brings back the zero volt level along with the positive or high level and the negative or low level so that there are three electrical levels. In this episode, we take a look at five line coding schemes for polar and bipolar signaling: NRZ-L, NRZ-I, RZ-AMI, Manchester coding, and differential Manchester coding.

One of the unipolar schemes, Non-Return-to-Zero or NRZ line coding, is also available in polar signaling, but appears as two different versions. Recall that unipolar NRZ represents a logic one with one of the two electrical levels and a logic zero with the other. Bipolar Non-Return-to-Zero Level or NRZ-L is similar in that it represents a logic one with one of the two non-zero voltage levels and a logic zero with the other voltage level.

An example of this is the RS-232 serial communications standard where the voltage level for a logic one is set to a negative level between negative three and negative fifteen volts while the voltage level for a logic zero is represented with a positive voltage level between three and fifteen volts. This polarity and the higher voltage levels allow for cable lengths of up to 15 meters in addition to providing significant immunity against electromagnetic interference.[1] Just like unipolar NRZ, however, there is still the potential for a loss of synchronization with long sequences of zeros or ones.

A second polar NRZ scheme, called Non-Return-to-Zero Inverted or NRZ-I, offers a solution to the synchronization problem without increasing bandwidth. In this case, the bit values are not identified by the level, rather they are identified by the timing of the level changes or transitions. A level change or transition at the beginning of a bit period identifies a logic one while no change identifies a logic zero. This process, where a bit's value is represented with the timing of the transitions rather than the value of the level itself, is referred to as differential coding. By using differential coding, NRZ-I offers the same data density as NRZ-L, yet adds a clock signal when there are ones in the data. Long sequences of zeros, however, still present a problem.

A quick note about the advantages of differential coding. First, when a signal has traveled for any distance, transitions are easier to detect than voltage levels. This makes differential coding more reliable. Second, since the data is described by the timing of the transitions rather than the voltage levels, the signal can be inverted and still be decoded to the same data stream. This could happen if the telecommunications installer has accidentally swapped wires. Lastly, since the data is represented by the transitions instead of the levels, any two levels can be used. This allows us to use polar signaling which eliminates voltage drift or bias and increases the signal's immunity to noise. In fact, the transitions could be represented with light or even magnetic polarity if needed.

Okay, back to the line codes. Another unipolar scheme that is carried over to bipolar is Return-to-Zero, and just like the unipolar implementation of RZ, the last half of every bit period equals zero volts. The first half of the bit period representing a logic one is a non-zero voltage while a logic zero is still represented with zero volts across the entire bit period.

The most common form of RZ is RZ-AMI, where AMI stands for Alternate Mark Inversion. The term Alternate Mark Inversion may make more sense if one understands the terms "mark" and "space". In serial communications, a mark often refers to a logic one whereas a space refers to a logic zero. Like unipolar RZ, a logic one in Alternate Mark Inversion is represented with a pulse for the first half of the bit period followed by a return to zero volts for the last half of the bit period. The difference is that the pulses in AMI alternate their polarity from the positive to the negative with each successive logic one. These alternating pulse levels prevent the voltage drift that occurs in other signaling schemes.

The last type of line coding we will discuss in this episode is Manchester code. Manchester code requires a level transition or edge in the middle of every bit period. Logic ones are defined by a transition or edge in one direction while logic zeros are identified by an edge in the other direction. Which edge is which? Well, it depends on which standard is followed. When following the standards published by IEEE, a logic one is represented with a rising edge in the middle of the bit period while a logic zero is represented with a falling edge. An edge will also occur between bit periods when two bits of the same value are next to each other in the data stream. Manchester code requires a bandwidth twice that of the bit rate since it is possible that the signal level changes twice as fast at the bit periods when transmitting successive ones or successive zeros.

There is a form of Manchester coding that uses differential coding. Like Manchester code, differential Manchester code always has a transition in the middle of a bit. Unlike Manchester code, there is no meaning to the direction of the transition in differential Manchester code. Logic ones and logic zeros are distinguished from one another by the absence or presence of a transition at the beginning of a bit period. If the bit period represents a logic one, there is no transition at the start of the bit period. If the represented bit is a logic zero, the bit period begins with a transition. That means that regardless of the voltage level "left over" by the previous bit, to transmit a logic one, leave the level the same for the first half of the bit period, then flip to the opposite voltage level at the mid-point of the bit time. To transmit a logic zero, invert the voltage level both at the beginning of the bit time and at the mid-point of the bit time.

In our next episode, we are going to look at how to improve data density using differential encoding. To do this, we will represent groups of bits with pulse patterns rather than individual bits. For transcripts, links, or other podcast notes, please check us out at intermation.com where you will also find links to our Instagram, Twitter, Facebook, and Pinterest pages. Until the next episode remember that while the scope of what makes a computer is immense, it's all just ones and zeros.

References:

1 – Circuit Digest, “RS232 Serial Communication Protocol: Basics, Working & Specifications,” https://circuitdigest.com/article/rs232-serial-communication-protocol-basics-specifications